主要内容

四、网络安全协议（续）

◆应用层安全协议

- ❖DNS安全协议DNSSEC
- ❖电子邮件的安全协议

五、Web安全

- ❖HTTP协议
- ❖针对服务器的攻击
- ❖针对客户端的攻击

DNS安全

域名系统DNS

- ◆ 域名
  - ❖ 域名是用来标识网络上的主机，它与主机的IP地址相对应，一台主机可以有多个域名。
- ◆ 层次树状结构的命名方法
  - ❖ 任何一个连接在因特网上的主机或路由器，都有一个惟一的层次结构的名字，即域名。
  - ❖ 域名的结构由若干个分量组成，各分量之间用点隔开：

    … .三级域名.二级域名.顶级域名
- ◆ 各分量分别代表不同级别的域名，域名系统是分级的分布式数据库系统，用来查找域名与IP地址的对应关系。
- ◆ 查询域名的应用程序叫解析器（resolver），存储域名与IP地址对应关系的服务器叫域名服务器。
- ◆ DNS报文传输层可采用TCP或UDP协议，端口号均为53号

域名的解析过程

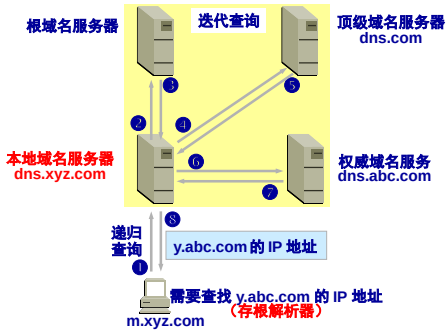
◆主机向本地域名服务器的查询通常采用递归查询。

- ❖如果主机所询问的本地域名服务器不知道被查询域名的 IP 地址，那么本地域名服务器就以 DNS 客户的身份，向其他根域名服务器继续发出查询请求报文。

◆本地域名服务器向根域名服务器的查询通常采用迭代查询。

- ❖当根域名服务器收到本地域名服务器的迭代查询请求报文时，要么给出所要查询的 IP 地址，要么告诉本地域名服务器：“你下一步应当向哪一个域名服务器进行查询”。然后让本地域名服务器进行后续的查询。

本地域名服务器采用迭代查询

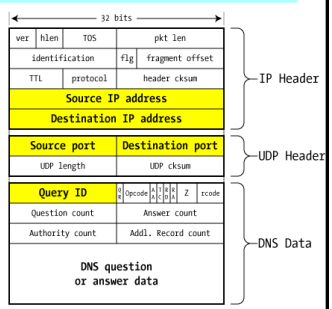


说明

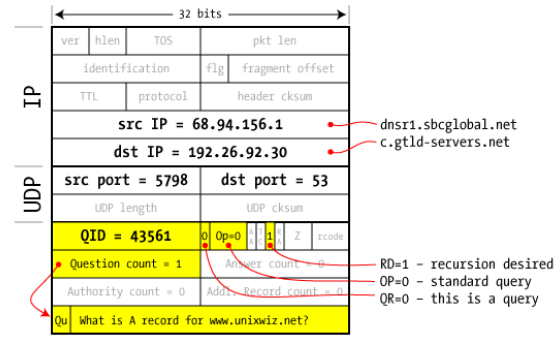
- 用户在浏览器中输入网站名称，浏览器使用存根解析器（操作系统内核）将网站域名转换为互联网协议 (IP) 地址。
- 存根解析器是一种DNS 客户端，将应用的 DNS 数据请求传递到递归解析器（复杂客户端）。
- 很多网络运营商运行递归解析器，以便处理设备通过其网络发送的 DNS 请求或查询。（有时，小型运营商和组织会对其他网络使用递归解析器，包括作为公众服务运行的递归解析器，如 Google Public DNS、OpenDNS 和 Quad9）
- 递归解析器将其 DNS 查询发送至多个不同的权威域名服务器

DNS 分组

- Query ID:
  - 16 bit random value (TXID)
  - Links response to query

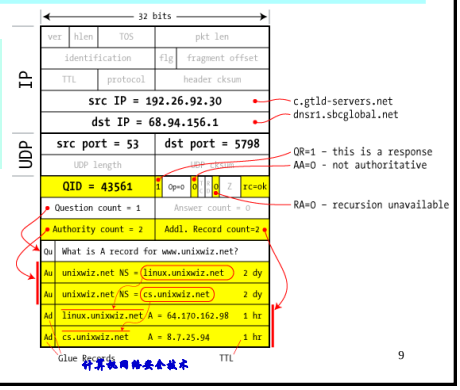


DNS解析器的查询请求



解析器响应

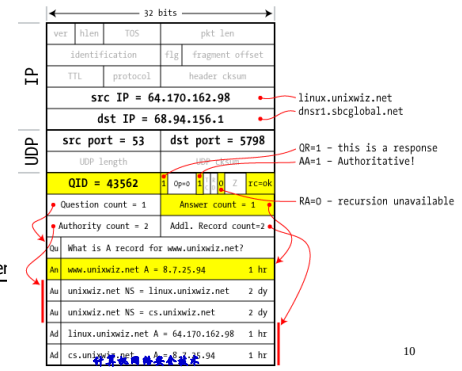
响应结果包含下一个域名服务器的IP地址(called “glue”)



权威响应（Authoritative response ）

范围检查（bailiwick checking）：缓存同一域的响应

final answer



DNS的高速缓存

- 每个域名服务器都维护一个高速缓存，存放最近用过的名字以及从何处获得名字映射信息的记录。
  - 减轻根域名服务器的负荷，使互联网上的 DNS 查询请求和回答报文的数量大为减少。
- 为保持高速缓存中的内容正确，域名服务器应为每项内容设置计时器，并处理超过合理时间的项（例如，每个项目只存放两天）。
- 当权威域名服务器回答一个查询请求时，在响应中都指明绑定有效存在的时间值。
  - 增加此时间值可减少网络开销，而减少此时间值可提高域名转换的准确性。

DNS的高速缓存（续）

- 一些操作系统维护本地的DNS缓存
  - Windows有本地DNS缓存
    - 浏览DNS缓存：ipconfig/displaydns
  - 很多发行版Linux没有，使用预定的域名服务器进行查询解析
  - 一些浏览器（如Firefox）也支持自己的DNS缓存
- 域名解析的无限循环
  - DNS查询响应中包括“Glue Record”，防止产生域名解析的递归依赖关系
    - 域名服务器和子域名服务器

例子

- 在Web浏览器中访问某个网址时
  - 优先访问浏览器缓存，如果未命中则访问OS缓存，最后再访问DNS服务器(一般是ISP提供)
- DNS服务器会递归式的查找域名记录，然后返回。
- DNS记录的ttl值(time to live，单位是秒)
  - 表示记录最大有效期是多少
  - 通常，操作系统OS缓存会参考ttl值，但是不完全等于ttl值，而浏览器DNS缓存的时间跟ttl值无关，每种浏览器都使用一个固定值。

域名解析命令

- Windows
  - 在控制台运行nslookup命令
- Linux用以下命令发送DNS请求
  - nslookup
  - dig

DNS的安全隐患

- 产生于上世纪 80 年代，当时互联网规模小得多，安全性并非首要设计考虑因素。
  - 用户/主机信任DNS查询结果
    - 将域名解析结果作为安全策略的基础
- 解析器无法验证响应的真实性（DNS欺骗 Spoofing）
  - 解析器无法轻易识别某一项查询的仿冒响应。攻击者很容易冒充看似来自权威服务器的响应
  - 在DNS之前将客户请求的信息劫持下来，并响应虚假数据给客户
- DNS响应被缓存 （缓存污染，缓存中毒）
  - 攻击者发送的仿冒 DNS 响应获得递归解析器的接受，则意味着递归解析器缓存污染成功。

DNS的安全隐患（续）

- DNS重定向
  - 将DNS域名查询重定向到恶意DNS服务器上
- 伪造数据
  - 用户与 DNS 服务器之间采用 UDP (用户数据报协议) 进行明文通信，容易被修改、伪造。
- 绕过防火墙
  - 防火墙不会限制对DNS的访问
  - 利用被控制的DNS服务器，绕过防火墙等其它安全设备的控制
- DNS的本身性能问题影响整个应用系统

一些针对DNS的攻击

- 拒绝服务攻击(Denial of Service)
  - 在短时间里，向DNS服务器发起成千上万的请求包，并保持“半开连接”状态，耗尽服务器资源，导致服务停止工作或机器宕机。
- 缓冲区漏洞溢出攻击(buffer Overflow)
  - 利用服务器存在的漏洞对DNS进行缓存区溢出攻击，造成服务停止或直接获取root密码。
- 域欺骗(Pharming)
  - 最早出现在2004年左右
  - 通过入侵DNS（Domain Name Server）的方式，将使用者导引到伪造的网站上，因此又称为DNS毒化（DNS Poisoning）
  - 被攻击的DNS服务器可能返回错误/恶意的响应
    - 例如：伪造Wifi热点导致的域名解析问题

DNSSEC

- DNSSEC（ Domain Name System Security Extensions， DNS安全扩展 ）
  - 由IETF提供的一系列DNS安全认证的机制

> “The Domain Name System (DNS) security extensions provide origin authentication and integrity assurance services for DNS data, including mechanisms for authenticated denial of existence of DNS data.”
>
> -RFC 4033

DNSSEC的部署

- 历史
  - 1997年第一个有关DNSSEC的标准RFC 2065
  - 1999年IETF对DNSSEC标准进行了修订RFC 2535
  - 2005年，一个可用的DNSSEC标准才制定出来（RFC 4033-4035）
  - 2008年，IETF又发布了一个NSEC3（RFC5155）标准，以提高DNSSEC隐私保护能力。
- 部署
  - 2010年7月，ICANN、Verisign以及NTIA等完成对DNS根区域进行签名部署
  - 主要DNS服务器支持
    - 如BIND, 微软，NSD, Unbound等

DNSSEC

- 一种可以认证应答信息真实性和完整性的机制
  - 利用公钥密码技术，域名解析服务器可以验证它所收到的应答(包括域名不存在的应答)是否来自于真实的服务器
  - 是否在传输过程中被篡改过
- 作用
  - 通过DNSSEC的部署，可以增强对DNS域名服务器的身份认证，防止DNS欺骗和缓存污染等攻击。
  - DNSSEC是实现DNS安全的重要组成部分

数字签名

- 权威域名服务器用自己的私有密钥对资源记录（Resource Record, RR）进行签名
- 解析服务器用权威服务器的公开密钥对收到的应答信息进行验证。
- 如果验证失败，表明这一报文可能是假冒的，或者在传输过程、缓存过程中被篡改了。

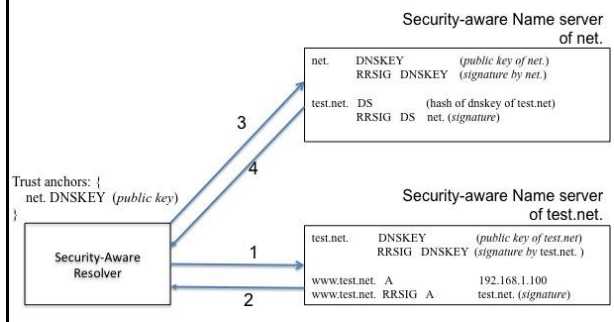
DNSSEC的特征

- 不改变分组格式
  - 保障DNS数据安全，而非信道安全
- 新的资源记录 (RRs)
  - 资源记录签名RRSIG :用于存储 DNS资源记录的签名信息
  - DNSKEY :存放用于检查 DNSSEC 签名的公钥
  - DS（Delegation Signer授权签名者）：用于DNSKEY验证过程，存储密钥标签，加密算法和对应的DNSKEY的摘要信息。
  - NSEC / NSEC3：存储和对应的所有者相邻的下一个资源记录；用于签名验证。

例：DNSSEC的工作原理

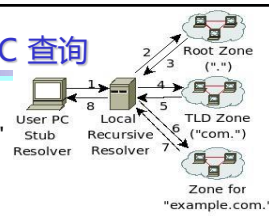
- 一个支持DNSSEC的解析器向支持DNSSEC的权威域名服务器（Security-Aware Name Server）请求域名 www.test.net
- 查询结果
  - 一个标准的A记录（包含IPv4地址）
  - 一个同名的RRSIG记录，其中包含test.net这个权威域的数字签名，它是用test.net的私有密钥来签名的。
- 为了验证签名，解析器再次向test.net的域名服务器查询响应的公开密钥，即名为test.net的DNSKEY类型的资源记录。
- 解析器就可以用该公钥验证www.test.net. 记录的真实性与完整性。

例：DNSSEC的工作原理



DNSSEC 查询

Query: "www.example.com A?"



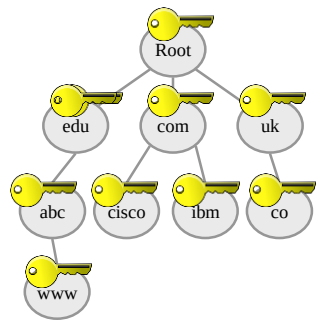
Reply	RRs in DNS Reply	Added by DNSSEC
3	"com. NS a.gtld.net" "a.gtld.net A 192.5.6.30"	"com. DS" "RRSIG(DS) by ."
5	"example.com. NS a.iana.net" "a.iana.net A 192.0.34.43"	"com. DNSKEY" "RRSIG(DNSKEY) by com." "example.com. DS" "RRSIG(DS) by com."
7	"www.example.com A 1.2.3.4"	"example.com DNSKEY" "RRSIG(DNSKEY) by example.com." "RRSIG(A) by example.com."
8	"www.example.com A 1.2.3.4"	Last Hop?

如何建立对域名服务器公钥的信任?

- 中间人攻击
  - 攻击者拦截流量，用自己的私钥伪造DNS响应记录，发送自己的公钥作为DNSKEY记录
- 防御
  - DNSSEC使用**信任链（chain of trust）**
- 数字签名验证
  - 每个DNS区域都有一个父区域，在层次结构中通过向上返回根域名服务器（受信任点）就能建立信任
  - 为了验证特定区域的公钥，客户端从其父区域请求指定签名者DS记录，该记录包含子区域公钥的散列
  - 客户端使用父域名服务器的DNSKEY来解密RRSIG记录，将结果与DS记录比较，最后再将DS记录与子域名服务器的DNSKEY比较
  - 这个过程一直进行，直到客户端已知道这是一个受信任的密钥，不再需要验证为止

DNSSEC的信任链

- PKI体系的映射
- 树结构组织



DNSSEC的问题

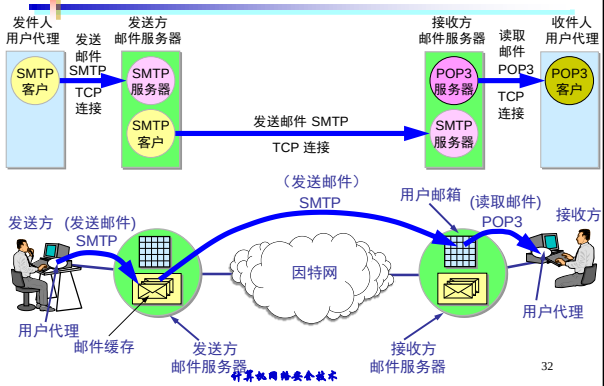
- 无法保证私密性
  - 仍基于 UDP，明文传输，只是增加一个数字签名
- 域名劫持发生时不能告诉用户真正的记录
  - 当用户的 DNS 被劫持时，用户通过检查 DNSSEC 签名，可以知道自己得到的并不是真正的解析结果，而是得到了一个被伪造的地址。但是，用户并不知道真正的解析结果是什么。
- 支持 DNSSEC 的递归服务器并不多
- 主流浏览器并没有原生对 DNSSEC 的支持

电子邮件的安全协议

电子邮件的标准协议

- 发送邮件的协议：SMTP
- 读取邮件的协议：POP3 和 IMAP
- MIME 在其邮件首部中说明了邮件的数据类型(如文本、声音、图像、视像等)，使用 MIME 可在邮件中同时传送多种类型的数据。

电子邮件系统的组成



用户代理 UA (User Agent)

- 用户代理 UA 是电子邮件客户端软件。
  - 撰写、显示、处理和通信。
- 邮件服务器
  - 发送和接收邮件，同时还要向发信人报告邮件传送的情况（已交付、被拒绝、丢失等）。
- 邮件服务器按照客户/服务器方式工作。
- 邮件服务器需要使用发送和读取两个不同的协议。

简单邮件传送协议 SMTP

- SMTP ：规定在两个相互通信的 SMTP 进程之间应如何交换信息。
- 客户/服务器方式
  - 负责发送邮件的 SMTP 进程就是 SMTP 客户，而负责接收邮件的 SMTP 进程就是 SMTP 服务器。
- SMTP
  - 规定了 14 条命令和 21 种应答信息。每条命令用 4 个字母组成，而每一种应答信息一般只有一行信息，由一个 3 位数字的代码开始，后面附上（也可不附上）很简单的文字说明。

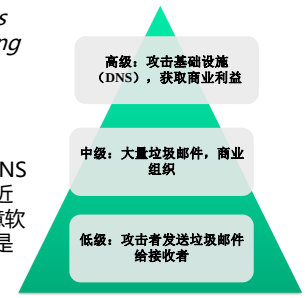
命令	描述
DATA	开始信息写作
EXPN<string>	验证给定的邮箱列表是否存在，扩充邮箱列表，也常被禁用
HELO<domain>	向服务器标识用户身份，返回邮件服务器身份
HELP<command>	查询服务器支持什么命令，返回命令中的信息
MAIL FROM<host>	在主机上初始化一个邮件会话
NOOP	无操作，服务器应响应OK
QUIT	终止邮件会话
RCPT TO<user>	标识单个的邮件接收人；常在MAIL命令后面可有多个rcpt to:
RSET	重置会话，当前传输被取消
SAML FROM<host>	发送邮件到用户终端和邮箱
SEND FROM<host>	发送邮件到用户终端
SOML FROM<host>	发送邮件到用户终端或邮箱
TURN	接收端和发送端交换角色
VRFY<user>	用于验证指定的用户/邮箱是否存在；由于安全方面的原因，服务器常禁止此命令

SMTP 通信的三个阶段

1. 连接建立：连接是在发送主机的 SMTP 客户和接收主机的 SMTP 服务器之间建立的。SMTP不使用中间的邮件服务器。
2. 邮件传送
3. 连接释放：邮件发送完毕后，SMTP 应释放 TCP 连接。

电子邮件的威胁

- RFC 4686（*Analysis of Threats Motivating DomainKeys Identified Mail*）
  - 电子邮件的安全威胁
  - DKIM的目标
- 美国安全研究机构SANS研究所的调查显示，近3/4的网络钓鱼、恶意软件和勒索软件攻击都是通过电子邮件开始的



针对电子邮箱的攻击事件

事件层次	针对性	攻击量	主要攻击目的	主要攻击手段
垃圾邮件	弱	大	发送商业推广信息	群发邮件
个体攻击	弱	大	盗号发送垃圾邮件 骗取个人财物	仿冒邮箱、钓鱼/鲸邮件、盗号木马、敲诈者木马、盗号、撞号
商业欺诈	强	中	骗取公司财物	仿冒邮箱、盗号、撞号
APT 攻击	强	小	情报窃取、系统破坏	邮件携带专用木马

垃圾邮件Spam

- Spam一词表示垃圾邮件。指未经请求而发送的电子邮件，如未经发件人请求而发送的商业广告或非法的电子邮件（1994.12）。
- 《中国互联网协会反垃圾邮件规范》的定义
  - （一）收件人事先没有提出要求或者同意接收的广告、电子刊物、各种形式的宣传品等宣传性的电子邮件；
  - （二）收件人无法拒收的电子邮件；
  - （三）隐藏发件人身份、地址、标题等信息的电子邮件；
  - （四）含有虚假的信息源、发件人、路由等信息的电子邮件。

垃圾邮件与“网络钓鱼”

- “网络钓鱼”
  - 通过大量发送声称来自于银行或其他知名机构的欺骗性垃圾邮件，意图引诱收信人给出敏感信息(如用户名、口令、帐号ID、ATMPIN码或信用卡详细信息)的一种攻击方式。
  - 最典型的网络钓鱼攻击将收信人引诱到一个通过精心设计与目标组织的网站非常相似的钓鱼网站上，并获取收信人在此网站上输入的个人敏感信息，通常这个攻击过程不会让受害者警觉。
  - 它是“社会工程攻击”的一种形式。
  - 充分利用了人们的心理弱点

电子邮件的安全协议

- PGP（Pretty Good Privacy）概述
  - 一个可提供机密性和认证的软件。通常用来加密和解密电子邮件。
- S/MIME （多用途网际邮件扩充协议 Secure Multipurpose Internet Mail Extensions）
  - 基于 MIME 标准， S/MIME 为电子消息应用程序提供如下加密安全服务：认证、完整性保护、鉴定及数据保密等。
- DKIM（Domain Keys Identified Mail）域名密钥识别邮件

电子邮件的安全性

- 机密性 confidentiality
  - protection from disclosure
- 认证 authentication
  - of sender of message
- 消息完整性 message integrity
  - protection from modification
- 防止否认 non-repudiation of origin
  - protection from denial by sender

PGP（Pretty Good Privacy ）

- 1991年，程序员Phil Zimmermann开发加密软件PGP（Pretty Good Privacy ），商业软件
  - 选择可用加密算法作为系统的构造模块，命令集合
  - 设计了程序、文档，并在Internet上公开
  - Network Associates公司提供了全兼容、低成本的商业版本
- 开源、多平台支持

- OpenPGP： RFC 4880定义
  - GnuPG（又称 GPG）是 OpenPGP 的一种开源实现
  - OpenPGP 网站 列出了一系列邮件加密解决方案，涵盖了各种桌面和移动操作系统
    网址：https://www.openpgp.org/software/

PGP的安全功能

- 数字签名
  - DSS/SHA或RSA/SHA
- 完整性
  - RSA、MD5
- 消息加密
  - CAST-128或IDEA或3DES + Diffie-Hellman或RSA
- 数据压缩
  - ZIP
- 邮件兼容
  - Radix 64：将任意二进制输入，转换成可打印的字符输出的编码技术。
- 数据分段
  - 大邮件自动分段并在接收时自动恢复。

OpenPGP 的工作原理

- 用户针对自己的电子邮件地址，生成一对密钥
  - 其中**公钥**是完全公开的，可以上传到公钥服务器，让通信的另一方通过邮件地址搜索到；也可以通过邮件等形式直接发送给对方。
  - 而**私钥**是保密的，只有用户自己知道。
- 通过“查找公钥”工具，搜索对方电子邮件地址或 key ID 查找。
- 安全问题
  - 如何保证公钥的可信性?
  - 滥用：垃圾邮件发送者开始发送经过加密和签名的邮件

S/MIME (Secure/Multipurpose Internet Mail Extensions)

安全/多用途网际邮件扩展

- MIME email的安全性增强
  - 早期的 RFC822定义电子邮件格式标准，最新版本RFC5322
  - MIME支持多种内容类型和正文中多个独立的部分
  - 编码：将二进制数据编码成文本格式
  - S/MIME：安全增强（security enhancements）
    - 发送经过**数字签名**和**加密**的邮件
- S/MIME支持多种邮件代理，例如
  - Outlook 2010 或更高版本
  - Web 上的 Outlook（以前称为 Outlook Web App）
  - Exchange ActiveSync (EAS)
- RFC 5751

S/MIME 功能

- 封装数据
  - 加密内容+相关密钥
- 签名数据
  - 签名的消息摘要+ 对消息和签名进行编码（base64）
- 透明-签名数据
  - 签名的消息摘要+明文消息 + 编码后的消息摘要签名
- 签名&封装数据
  - 嵌套的签名和加密实体

S/MIME 加密算法

- 数字签名：如RSA
- 哈希（散列）函数: SHA-1 & MD5
- 会话密钥加密：如RSA
- 消息加密: AES, Triple-DES, RC2/40 等
- MAC: HMAC with SHA-1

S/MIME 消息

- 签名和加密
  - S/MIME secures a MIME entity with a signature, encryption, or both
- 内容类型:
  - 加密的数据
  - 签名的数据
  - 透明-签名的数据
  - 认证请求
  - 仅认证消息

S/MIME 证书管理

- S/MIME 使用 X.509 v3证书
- 混合方式
  - 严格的 X.509 CA证书 体系 & PGP的基于web的信任方式
- 每个客户端拥有可信CA证书列表
- 以及自己的公钥/私钥对及其证书
- 证书必须有可信CA签名
  - 例如，Verisign认证服务

S/MIME 增强的安全服务

- 三种增强的安全服务:
  - 签收：signed receipts
  - 安全标记：security labels
  - 安全邮件列表

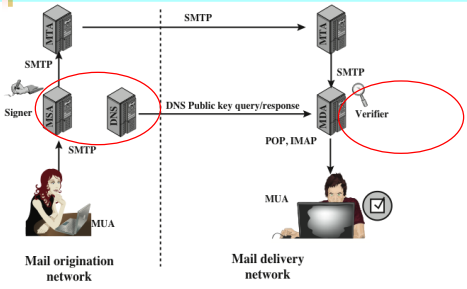
域名密钥识别邮件

- DKIM（Domain Keys Identified Mail）
  - 用密码学方法签名电子邮件的规范，支持一个签名域对邮件流中的某个邮件签名
    - 对邮件头及正文进行签名
  - 邮件接收方可以验证签名
    - 查询签名者的域检索相应的公钥，以便确认消息是否由拥有该签名域私钥的一方签名
- 标准：rfc4871，rfc5762，rfc8301
- 应用：邮件提供商gmail，Yahoo，网易、QQMail等，多家ISP(Internet Service Providers)
- 防止“钓鱼邮件”

DKIM工作原理

- DKIM签名是先对内容（BODY）部分计算hash值，然后把这个BODY HASH放到HEADER里面，再对头部做签名。
- 选择头部常用字段进行签名
  - 例如，FROM则是必须被签名(rfc4871 5.5 Recommended Signature Content)
- 最后在邮件头中增加一个DKIM-Signature头用于记录签名信息。
- 接收方则通过DNS查询得到公开密钥后进行验证
  - 验证不通过，则认为是垃圾邮件
- DKIM不仅仅可以防止垃圾邮件，还可以防止邮件内容被篡改。

DKIM部署示例



DNS = domain name system
MDA = mail delivery agent
MSA = mail submission agent
MTA = message transfer agent
MUA = message user agent

Web安全

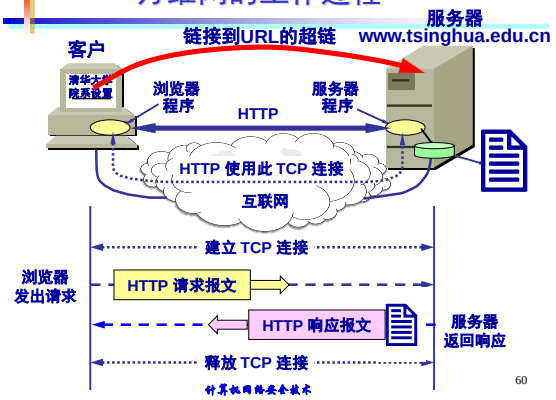
HTTP协议基础

参考：谢希仁，《计算机网络》（第七版）

超文本传送协议 HTTP

- 浏览器和Web服务器之间的应用层协议
  - HTTP 协议用来传送超文本的链接信息
- HTTP 是 面 向 事 务 的 (transaction-oriented)应用层协议
  - 它是万维网上能够可靠地交换文件（包括文本、声音、图像等各种多媒体文件）的重要基础。

万维网的工作过程



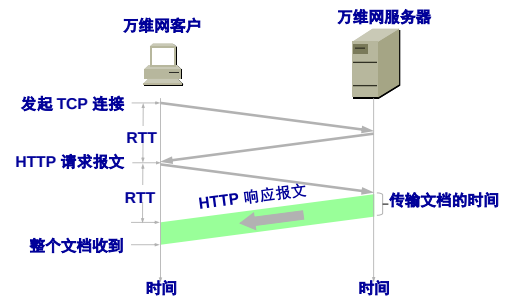
HTTP协议

- 1996年，HTTP1.0
  - 每次请求都需要与服务器建立一个TCP连接，服务器处理完成后立即断开TCP连接（无连接），服务器不跟踪每个客户端也不记录过去的请求（无状态）
  - 使用keep-alive参数告知服务器建立一个长连接
- 1999年，HTTP1.1（广泛使用）
  - HTTP1.1默认支持长连接：使用Connection: keep-alive
  - 支持只发送header信息(不带任何body信息)
  - 还提供了与身份认证、状态管理和Cache缓存等机制相关的请求头和响应头
- 2015年，HTTP2.0
  - 支持多路复用，解决了1.1的长连接会遇到阻塞的问题
  - 首部数据压缩；支持传输二进制信息
  - 服务端主动推送内容到客户端缓存，避免往返时延

HTTP 1.0

- HTTP 是面向事务的客户服务器协议。
- HTTP 1.0 协议是无状态的 (stateless)。
- HTTP 协议本身也是无连接的，虽然它使用了面向连接的 TCP 向上提供的服务。

请求一个万维网文档所需的时间



持续连接

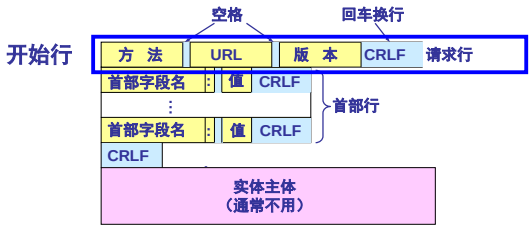
- HTTP/1.1 协议使用持续连接 (persistent connection)。
- Web服务器在发送响应后仍然在一段时间内保持这条连接，使同一个客户（浏览器）和该服务器可以继续在这条连接上传送后续的 HTTP 请求报文和响应报文。
- 非流水线方式：客户在收到前一个响应后才能发出下一个请求。
  - 开销节省了建立 TCP 连接所需的一个 RTT 时间。但服务器在发送完一个对象后，其 TCP 连接就处于空闲状态，浪费了服务器资源。
- 流水线方式：客户在收到 HTTP 的响应报文之前就能够接着发送新的请求报文。
  - 一个接一个的请求报文到达服务器后，服务器就可连续发回响应报文。使用流水线方式时，客户访问所有的对象只需花费一个RTT时间，使 TCP 连接中的空闲时间减少，提高了下载文档效率。

HTTP 的报文结构

HTTP 有两类报文：

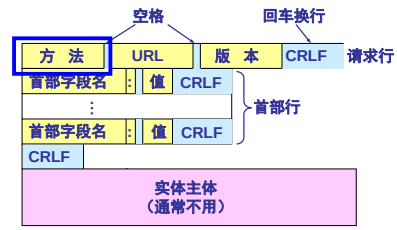
- 请求报文——从客户向服务器发送请求报文。
- 响应报文——从服务器到客户的回答。
- 由于 HTTP 是面向正文的 (text-oriented)，因此在报文中的每一个字段都是一些 ASCII 码串，因而每个字段的长度都是不确定的。

HTTP 的报文结构（请求报文）



报文由三个部分组成，即开始行、首部行和实体主体。在请求报文中，开始行就是请求行。

HTTP 的报文结构（请求报文）

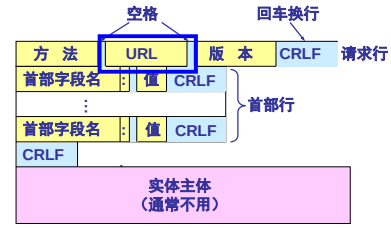


“方法”是面向对象技术中使用的专门名词。所谓“方法”就是对所请求的对象进行的操作，因此这些方法实际上也就是一些命令。因此，请求报文的类型是由它所采用的方法决定的。

HTTP 请求报文的一些方法

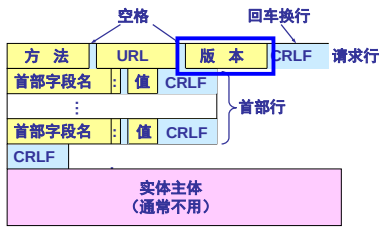
方法（操作）	意义
OPTION	请求一些选项的信息
GET	请求读取由 URL所标志的信息
HEAD	请求读取由 URL所标志的信息的首部
POST	给服务器添加信息（表单数据提交）
PUT	在指明的 URL下存储一个文档
DELETE	删除指明的 URL所标志的资源
TRACE	用来进行环回测试的请求报文
CONNECT	用于代理服务器

HTTP 的报文结构（请求报文）



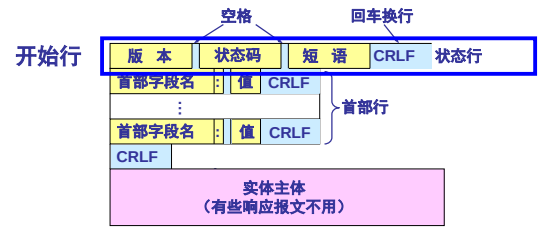
“URL”是所请求的资源的 URL。

HTTP 的报文结构（请求报文）



“版本”是 HTTP 的版本。

HTTP 的报文结构（响应报文）



响应报文的开始行是状态行。
状态行包括三项内容，即 HTTP 的版本，状态码，以及解释状态码的简单短语。

状态码

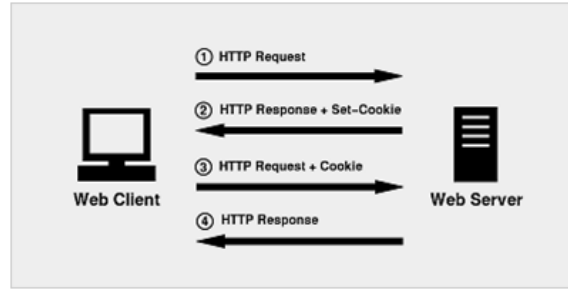
- 1xx 表示通知信息的，如请求收到了或正在进行处理。
- 2xx 表示成功，如接受或知道了。
- 3xx 表示重定向，表示要完成请求还必须采取进一步的行动。
- 4xx 表示客户的差错，如请求中有错误的语法或不能完成。
- 5xx 表示服务器的差错，如服务器失效无法完成请求。

在服务器上存放用户的信息

- Web站点使用 Cookie 来跟踪用户。
- Cookie 表示在 HTTP 服务器和客户之间传递的状态信息。
- 使用 Cookie 的网站服务器为用户产生一个唯一的识别码。利用此识别码，网站就能够跟踪该用户在该网站的活动。

Cookie技术：客户端会话保持

- Cookie是通过HTTP协议扩展实现的，即在HTTP请求头里面增加Cookie字段，用于存储客户端信息
  1. 客户端向Web服务器发起HTTP请求；
  2. 服务器在返回响应时，在HTTP响应头中设置Set-Cookie字段，该字段存储客户端信息和状态；
  3. 客户端解析服务器HTTP响应报头中的Set-Cookie字段，并根据其生命周期创建不同类型的Cookie（例如持久化Cookie就会在客户端硬盘上创建Cookie文件）。之后客户端每次发送HTTP请求报文时，都会在请求报头中增加Cookie字段。
  4. 服务器接收客户端的HTTP请求之后，从报文头中取出Cookie数据，来校验客户端状态或身份信息



Session技术：服务器端会话保持

- Session保存在服务器端，通过Session的唯一标识来区分不同的客户端。
  1. 客户端向服务器发起请求；
  2. 服务器端创建Session对象，程序可以操作这个对象，将客户端身份信息等以key-value形式写入Session。Session对象有一个唯一ID号，例如Tomcat产生的session对象唯一标识是JSESSIONID；
  3. 服务器端响应报文的报头中会携带Session ID；
  4. 客户端在以后的请求报头中都会携带Session ID，服务器端根据这个ID号判断客户端身份。

Web安全

Web安全

- 基础设施服务的安全问题
  - Web服务企业信息化
  - 社交平台： Web2.0、社交网络、微博
  - 基于 Web 的攻击和破坏安全风险也剧增
- 两个方面
  - Web服务器安全
    - Web服务器本身的安全和软件配置
    - 应用程序的安全：在Web服务器上运行的Java、ActiveX、PHP、ASP代码的安全
  - Web客户端安全

Web服务器攻击类型

- Web服务器攻击利用Web服务器软件和配置中常见的漏洞
  - 缓冲区溢出
  - 目录遍历
  - 脚本权限
  - 目录浏览
  - Web服务器软件默认安装的示例代码
  - Web服务器上运行的其他软件中的漏洞，例如SQL数据库软件

Web服务器的保护

- 给Web服务器或守护进程配置能够使它正常运行的最小权限
- 安装最新的安全补丁，关注漏洞的最新动态
- 删除默认示例
- 删除不需要的应用程序，安全配置同一台计算机上的其他网络服务，安装操作系统最新安全补丁
- 确保只给包含需要执行脚本的目录可运行权限
- 在Web服务器上每个目录中都提供一个index.html文件，避免目录浏览

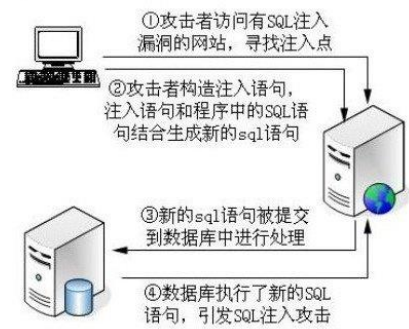
Web应用程序安全

- 漏洞攻击者利用不安全的Web应用程序来危害整个服务器，或破坏一个网站。
- 主要WEB 漏洞（Vulnerabilities）
  - SQL 注入攻击（SQL Injection）
  - 跨站域请求伪造(CSRF – Cross-site request forgery)
  - 跨站脚本攻击(XSS – Cross-site scripting)

SQL 注入攻击

- 产生原因
  - SQL注入往往是在编写包含用户输入的动态数据库查询时产生的
  - 程序员在Web开发时，没有过滤特殊字符，绑定变量，导致攻击者可以通过SQL灵活多变的语法，构造特殊语句，达成目的，或者通过系统报错，返回对自己有用的信息。
    - 浏览器将错误（恶意）输入发送给服务器
    - 缺乏必要的输入合法性检查导致错误的 SQL 查询
- 危害
  - 整个数据库的信息都能被读取或篡改
  - 获得更多的包括管理员的权限

SQL 注入攻击过程



例子

- 填好正确的用户名(tarena)和密码(admin)后，点击提交
  - 根据我们提交的用户名和密码被合成到SQL查询语句：
  - select * from users where username='tarena' and password=md5('admin')
- 对于有SQL注入漏洞的网站来说，只要构造个特殊的“字符串”，照样能够成功登录
  - 例如，在用户名输入框中输入:' or 1=1#,密码随便输入，这时候的合成后的SQL查询语句为：
  - select * from users where username='' or 1=1#' and password=md5('')
  - “#”在mysql中是注释符，等价于：

    select * from users where username='' or 1=1

因为1=1永远都是成立的，即where子句总是为真，将该sql进一步简化之后，等价如下select语句：

select * from users

防御SQL 注入攻击

- 用户输入验证
  - 对用户的输入进行校验，可以通过正则表达式，或限制长度，对单引号和双"-"进行转换等。
  - 不要使用动态拼装SQL，可以使用参数化的SQL或者直接使用存储过程进行数据查询存取。
- 权限管理
  - 不要使用管理员权限的数据库连接，为每个应用使用单独的权限有限的数据库连接。
- 其他
  - 不要把机密信息按明文形式存放。
  - 应用的异常信息应该给出尽可能少的提示，最好使用自定义的错误信息对原始错误信息进行包装，把异常信息存放在独立的表中

作业

- 试比较普通DNS请求应答方法和DNSSEC的应答和身份验证方法。
- DNSSEC的信任链是如何建立的?
- 安全电子邮件协议提供哪些安全机制?选择一个协议举例说明
- 举例说明Web服务器的三种安全漏洞